Soap Webservices (JAX-WS):

==============================

An service over the web is called as WebService.

JAX-WS: Java API for XML Web Services

WebServices History:

========================

Lets take an exmaple of School application.

1. School A implmented on e web application to store the student address details.

2. Now School B want the same kind of approach you have use for student address details. School B requested address module code to put in their end.

3. Now School A will extract the Adress module code as Jar and hand over it to the School B.

4. In future School A did few more changes to the Address module again School A will extract the Adress module code as Jar and hand over it to the School B.

To over come the above issues WebService come into picture. Webservices will do the following things,

Web service say deploy your application in application and give the service details to the consumer. I will help them to get your details.

If you are writing any web service we should register it in UDDI to make it accessible for others.

Lets take School A code written in Java nd School B code written in dotnet. In this we must required a common languae understand by any framework(Java/.Net/c/c++). To solve this common language XML came into the picture.

To carry the xml data to consumer from producer we need an protocol that is SOAP was introduced.

To convert the JAVA/.NET/C++/C specific code to SOAP we need an interface. Here SEI(Service endpoint Interface) came and it will work specific to source lanaguage and convert it to SOAP.

UDDI (Universal Description, Discovery and Integration)

===

This is the place will register our webservice to make it accessible for required consumers.

SEI (Service End Point Interface):

====================================

To convert the JAVA/.NET/C++/C specific code to SOAP we need an interface. Here SEI(Service endpoint Interface) came and it will work specific to source lanaguage and convert it to SOAP.

WSDL (Web Service Description Language):

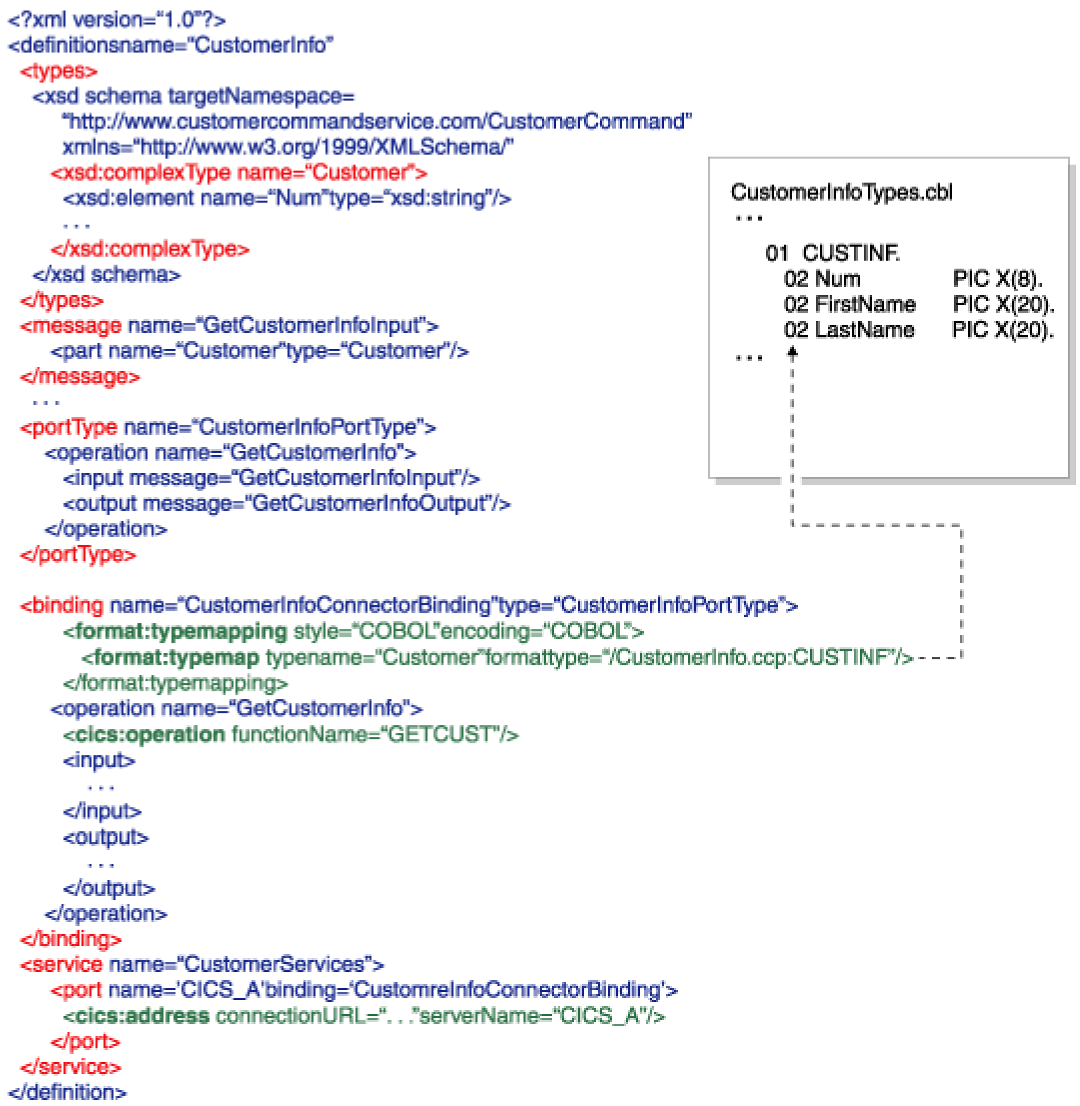
==

WSDL is an acronym for Web Services Description Language.

WSDL is a xml document containing information about web services such as method name, method parameter and how to access it.

WSDL is a part of UDDI. It acts as a interface between web service applications.

WSDL parameters:



```
<?xml version="1.0"?>
<definitionsname="CustomerInfo"
  <types>
    <xsd schema targetNamespace=
        "http://www.customercommandservice.com/CustomerCommand"
        xmlns="http://www.w3.org/1999/XMLSchema/"
      <xsd:complexType name="Customer">
        <xsd:element name="Num"type="xsd:string"/>
        . . .
      </xsd:complexType>
    </xsd schema>
  </types>
  <message name="GetCustomerInfoInput">
      <part name="Customer"type="Customer"/>
  </message>
  . . .
  <portType name="CustomerInfoPortType">
     <operation name="GetCustomerInfo">
        <input message="GetCustomerInfoInput"/>
        <output message="GetCustomerInfoOutput"/>
     </operation>
  </portType>

  <binding name="CustomerInfoConnectorBinding"type="CustomerInfoPortType">
        <format:typemapping style="COBOL"encoding="COBOL">
           <format:typemap typename="Customer"formattype="/CustomerInfo.ccp:CUSTINF"/>
        </format:typemapping>
      <operation name="GetCustomerInfo">
        <cics:operation functionName="GETCUST"/>
        <input>
           . . .
        </input>
        <output>
           . . .
        </output>
     </operation>
  </binding>
  <service name="CustomerServices">
      <port name='CICS_A'binding='CustomreInfoConnectorBinding'>
        <cics:address connectionURL=". . ."serverName="CICS_A"/>
      </port>
  </service>
</definition>
```

```
CustomerInfoTypes.cbl
...
   01  CUSTINF.
     02 Num          PIC X(8).
     02 FirstName    PIC X(20).
     02 LastName     PIC X(20).
...
```

definitions:

----

t is the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.

types:

----

its specifies the data types we are going to use in messages it may be a primitive or custom types.

messages:

Defines the data elements for each operation.

portType:

Describes the operations that can be performed and the messages involved.

binding:

It is the concrete protocol and data formats for the operations and messages defined for a particular port type.

Port:

Associates the binding with the URI http://www.examples.com/SayHello/ where the running service can be accessed.

service:

It is a collection of related end-points encompassing the service definitions in the file; the services map the binding to the port and include any extensibility definitions.

https://www.tutorialspoint.com/wsdl/wsdl_example.htm

Command to generate the Stubs from wsdl url:

==

```
wsimport -keep -s src  https://www.google.com?WSDL
```

=> keep -> will get the .java files

=> s -> To save the generated java files in specified location src

Command to generate the WSDL from XSD:

==

Conversion from XSD to WSDL can be done through Apache CXF.

xsd2wsdl xsdurl

SOAP (Simple Object Access Protocol):

==

SOAP is an acronym for Simple Object Access Protocol.

SOAP is a XML-based protocol for accessing web services.

SOAP is a W3C recommendation for communication between applications.

SOAP is XML based, so it is platform independent and language independent. In other words, it can be used with Java, .Net or PHP language on any platform.

Advantages of Soap Web Services:

--

WS Security: SOAP defines its own security known as WS Security.

Language and Platform independent: SOAP web services can be written in any programming language and executed in any platform.

Disadvantages of Soap Web Services:

--

Slow: SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications. So it is slow and consumes more bandwidth and resource.

WSDL dependent: SOAP uses WSDL and doesn't have any other mechanism to discover the service.

Bigger learning curve

SOAP Building Blocks:

Envelope:

An Envelope element that identifies the XML document as a SOAP message – This is the containing part of the SOAP message and is used to encapsulate all the details in the SOAP message. This is the root element in the SOAP message.

Header:

A Header element that contains header information – The header element can contain information such as authentication credentials which can be used by the calling application.

Body:

A Body element that contains call and response information – This element is what contains the actual data which needs to be sent between the web service and the calling application.

fault (default):

When a request is made to a SOAP web service, the response returned can be of either 2 forms which are a successful response or an error response. When a success is generated, the response from the server will always be a SOAP message. But if SOAP faults are generated, they are returned as "HTTP 500" errors.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">

    <SOAP-ENV:Body>

      <SOAP-ENV:Fault>

      <faultcode xsi:type="xsd:string">SOAP-ENV:Client</faultcode>

     <faultstring xsi:type="xsd:string">

        Failed to locate method (GetTutorialID) in class (GetTutorial)

      </faultstring>
    </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Best practices for WebServices:

================================

1. Always write the Interface and implementation class. As we put the @WebService annotation on the interface and WSDL generaeted from the interface if we share it will amny consumers webservice api's will be common for all. We can change the logics in Implemenations classes it wont effect the consumers.

2. Always good to write the WSDL first then generate the code.

WebServices Approaches:

===========================

Top Down Approach:

===================

In top-down approach, first you design the implementation of the Web service by creating a WSDL file. You can then create the Web service skeleton Java classes from the wsdl, and add the required code.

Bottom Up Approach:

=========================

In Bottom up Approach first you write the java classes for the web service and then create the WSDL file and publish the web service.

Although bottom-up Web service development may be faster and easier, the top-down approach is the recommended way of creating a Web service.

Webservice Annotations:

============================

@Webservice: annotation makes this interface/class a web service interface. The implementing class should also be annotated with @WebService.

@Webmethod: Annotation denotes that this method will be published and used as a web service.The method must be public as per javadoc.

SOAPBinding(style = Style.RPC): defines a SOAP styles web service with binding style as RPC

SOAPBinding(style = Style.DOCUMENT): defines a SOAP styles web service with binding style as DOCUMENT.

QName: is used to create qualified name for HelloWorldServerImplService which maps to the name generated in wsdl.